Hashing of Data Bases Based on Indirect Observations of Hamming Distances *

V.B.Balakirsky
The author is with the Department of Information Theory,
University of Lund, P.O. Box 118, S-221 00 Lund, Sweden
on leave from The Data Security Association "Confident",
St.-Petersburg, Russia
E-mails:
volodya@dit.lth.se,
vbal@newton.spb.su,
vbal@mathematik.uni-bielefeld.de

Abstract. We describe hashing of data bases as a problem of information and coding theory. It is shown that the triangle inequality for the Hamming distances between binary vectors may essentially decrease the computational efforts of a search for a pattern in the data base. Introduction of the Lee distance in the space, which consists of the Hamming distances, leads to a new metric space where the triangle inequality can be effectively used.

Index. Hashing, Searching for Patterns, Coding, Decoding

*The work was supported by a Scholarship from the Swedish Institute, Stockholm, Sweden.

1 Introduction

One of important problems in computer science can be represented as follows : we have given a collection of items, and we wish to store these items and upon demand retrieve the items whose key-values match given key-values. A particular approach to the storage and retrieval problem is known as hashing when we use the key-value of an item to compute an address for the storage of the item. Since the mapping between keys and addresses is not one-to-one, the events when different keys have the same address may take place; these events are known as collisions, and their resolution is of the main interest for a hashing scheme [1-4].

Similar ideas form the basis for the procedures which are referred to as external hashing [5]. It is known that the cost per storage increases as the access time decreases. Main memories are usually of the random access time. Since their size is limited with the cost requirements, the data bases are stored in the secondary memory with a rather slow access [6]. The hashing technique can essentially reduce the total number of accesses required if the values of a hash function applied to the keys of each item of the data base are stored in the main memory. To find the item with a certain key we calculate the value of the hash function for that key and access only the items whose key-values have the same value of the hash function. In the present paper, we address the external hashing and formulate the following problem (to simplify formalization, the item and its key value are associated) : we have given a collection of items which are binary vectors of the same length stored on the external memory, and a part of random access memory (RAM) that can be filled with the values of a hash function corresponding to each item. For a given pattern, which is a binary vector of the same length as the items, we must find all the items located at the Hamming distance less than a fixed threshold value.

We develop the external hashing in two directions. First, the value of the threshold can be positive, and it puts some restrictions on the hash functions allowed since we want to get information about the Hamming distances between binary vectors based on the comparison of the values of the hash function corresponding to these vectors. To make it possible, we use the metric properties of the Hamming space and extend some approaches by V.Koshelev [7] to the problem of finding the neighbouring points in random digital fields (these results are briefly stated in Section 3). Second, we represent the hashing problem as a special case of the data compression problem and use some results of coding theory which lead to the hashing procedure with the minimal possible average number of accesses to the external memory when the threshold is equal to zero (the optimality of the algorithm for positive thresholds is an open problem).

The basic quantity, we are interested in, is the number of accesses to the external memory after the use of the data stored in RAM. This quantity should be minimized in the average over all possible data bases and all posible patterns. It means that the hashing procedure must be fixed when we are not given a particular data base, but a probability distribution on data bases. Section 3 contains an example when conventional hashing schemes may lead to essentially different results for different probability distributions, and the best result is attained when each item is an independently chosen binary vector. Our hashing scheme gives the same result (in asymptotical sence) for any probability distribution.

The hashing procedure, introduced in the paper, is parametrized by a binary matrix, called a code. A random coding method [8], developed for the hashing scheme, leads to the theorem of an existence type, i.e., we claim that there exists a code such that the average number of accesses to the external memory is upperbounded by some expression. If the size of the data base exponentially grows with the length of the items, then the size of a code grows slower than a linear function of that length. Therefore, we can find a good code much easier compared to a similar problem in coding theory where the size of the code is usually assumed to be an exponential function of the code length.

¿From a formal point of view, the basic idea of the paper can be represented as an introduction of the Lee metric over the space consisting of the Hamming distances between binary vectors, since it gives an opportunity to organize an effective procedure of selecting the vectors which can be close to each other in the Hamming sence. Some properties of coverings of the Hamming space in the Lee metric, developed in the paper, can be usefull for a universal data compression with a fidely criterion [8,9].

The paper is organized as follows. In Section 2, we formulate the mathematical problem and the main result. In Section 3, we discuss possible indirect estimations of the Hamming distance between binary vectors. The hashing algorithm, based on these estimations, is described in Section 4. In Section 5, we prove the main result. Section 6 contains an example of the hashing procedure applied to the data base consisting of 50 names of the states of U.S.A. Some illustrations of hashing methods and proofs of auxiliary results are given in Appendix.

2 Mathematical Model for Search Problem. Formulation of Results

Let $\mathbf{X}$ be a binary matrix of size $M \times L$ that will be referred to as the data base (DB) and let $\mathbf{x} \in \{0,1\}^L$ be a vector that will be referred to as *pattern.* Let $\mathbf{x}_1, ..., \mathbf{x}_M$ be the rows of the matrix $\mathbf{X}$ that will be referred to as *records of the DB*. The problem is to form the set

$$J_T(\mathbf{x}, \mathbf{X}) = \{ \ j : d_H(\mathbf{x}, \mathbf{x}_j) \leq T \ \} \tag{2.1}$$

for given $\mathbf{x}, \mathbf{X}$, and *a threshold value* T, where $d_H(\cdot, \cdot)$ denotes the Hamming distance.

We suppose that the DB is stored on the external memory of the computer, but a preprocessing of the DB is possible, and there are Ml bits of RAM available to store the results. The operations should be defined by a function f, which maps the binary vectors of length L to the binary vectors of length l. The l-tuples $f(\mathbf{x}_1), ..., f(\mathbf{x}_M)$ contain information about $\mathbf{x}_1, ..., \mathbf{x}_M$, which may be used for selecting the numbers of records that are close to a given pattern using the Hamming distance as the measure of closeness. We can write the results of this selection via the values of a Boolean function φ_T, which is defined for all pairs $(\mathbf{x}, f(\mathbf{x}_1)), ..., (\mathbf{x}, f(\mathbf{x}_M))$ and equal to 1 for the pairs $(\mathbf{x}, f(\mathbf{x}_j)), j \in J_T(\mathbf{x}, \mathbf{X})$. Thus, we represent *hashing* of the DB as an assignment of the functions :

$$\begin{array}{rclcll} f & : & \mathbf{x}_j & \in & \{0,1\}^L & \longrightarrow & \{0,1\}^l, \\ \varphi_T & : & (\mathbf{x}, f(\mathbf{x}_j)) & \in & \{0,1\}^L \times \{0,1\}^l & \longrightarrow & \{0,1\}. \end{array} \tag{2.2}$$

such that the following statement :

$$d_H(\mathbf{x}, \mathbf{x}_j) \leq T \ \Rightarrow \ \varphi_T(\mathbf{x}, f(\mathbf{x}_j)) = 1 \tag{2.3}$$

is valid for all vectors $\mathbf{x}, \mathbf{x}_j \in \{0,1\}^L$. The functions f and φ_T will be referred to as *encoding* and *decoding*, respectively. In general, the function f

may depend on T, but we restrict our attention to the case when only the decoding depends on T.

Remarks. In computer science, the function f is referred to as a hash function, and hashing as an assignement of such a function. This situation follows from the fact that if $T = 0$ then $\varphi_0 = f^{-1}$. In our case, T can be positive, and it is not evident how to assign φ_T. Note that the similar problems arise in noiseless data compression and in data compression with a fidely criterion [8,9].

Using (2.3), we conclude that

$$J_T(\mathbf{x}, \mathbf{X}) \subseteq J(\mathbf{x}, \mathbf{X}|f, \varphi_T), \tag{2.4}$$

where

$$J(\mathbf{x}, \mathbf{X}|f, \varphi_T) = \{\ j : \varphi_T(\mathbf{x}, f(\mathbf{x}_j)) = 1\ \}. \tag{2.5}$$

Therefore, the problem can be solved in two steps :

1) to calculate $\varphi_T(\mathbf{x}, f(\mathbf{x}_1)), ..., \varphi_T(\mathbf{x}, f(\mathbf{x}_M))$ using the data, which are stored in RAM, and to construct the set $J(\mathbf{x}, \mathbf{X}|f, \varphi_T)$;

2) to transmit the records $\mathbf{x}_j, j \in J(\mathbf{x}, \mathbf{X}|f, \varphi_T)$ from the external memory to RAM and to construct the set $J_T(\mathbf{x}, \mathbf{X})$ after calculation of the Hamming distances $d_H(\mathbf{x}, \mathbf{x}_j)$ and their comparison with the threshold value T.

The complexity of the procedure is determined by the number of computations at the first and at the second step. However, supposing that the transmission plays the determining role, we characterize effectiveness of the algorithm by the quantity

$$\begin{aligned} N_T(\mathbf{x}, \mathbf{X}|f, \varphi_T) &= |\ J(\mathbf{x}, \mathbf{X}|f, \varphi_T) - J_T(\mathbf{x}, \mathbf{X})\ | \\ &= \sum_{j=1}^{M} \chi\{\ d_H(\mathbf{x}, \mathbf{x}_j) > T\ \} \cdot \chi\{\ \varphi_T(\mathbf{x}, f(\mathbf{x}_j)) = 1\ \}, \end{aligned} \tag{2.6}$$

Hereafter, χ denotes the indicator function : $\chi\{\beta\} = 1$ if the statement β is true, and 0 otherwise. These transmissions can be interpreted as 'extra' transmissions.

Suppose that the matrix $\mathbf{X}$ is chosen as a DB with the probability $P(\mathbf{X})$, and the vector $\mathbf{x} \in \{0,1\}^L$ appears as a pattern with the probability $p(\mathbf{x}|\mathbf{X})$. Then

$$N_T(p, P|f, \varphi_T) = \sum_{\mathbf{x}} \sum_{\mathbf{X}} p(\mathbf{x}|\mathbf{X}) \cdot P(\mathbf{X}) \cdot N_T(\mathbf{x}, \mathbf{X}|f, \varphi_T) \tag{2.7}$$

is the average number of "extra" transmissions. We set the mathematical problem as the minimization of (2.7) for given p and P over all admissible pairs (f, φ_T). In other words, it is required to evaluate the following function:

$$N_T(p, P) = \min_{f,\varphi_T} N_T(p, P|f, \varphi_T), \tag{2.8}$$

where the minimum is taken over all pairs (f, φ_T) satisfying (2.2) and (2.3). The aim of the paper is to propose an encoding and decoding method providing a sufficiently strong upper bound on the expression at the right hand side of (2.7) when T is small as compared to $L^{1/2}$. Obviously, the bound, we will obtain, is also an upper bound on $N_T(p, P)$.

Note that the same problem is also possible when the DB is fixed. However, it is included into our considerations because there are no restrictions on P, which may be defined as 0 for all DBs that do not coincide with a fixed one. The problem, we have formulated, is more general since, in practice, contents of the DB may be changed during operations.

Theorem. Let

$$M_t(\mathbf{X}|\mathbf{x}) = \sum_{j=1}^{M} \chi\{ \ d_H(\mathbf{x}, \mathbf{x}_j) = t \ \} \tag{2.9}$$

be the number of records located at the Hamming distance t from a given pattern $\mathbf{x}$, where $t = 0, ..., L$. Then, for any even q, there exist an encoding f^* and decoding φ_T^*, satisfying (2.2) and (2.3), such that

$$N_T(p, P|f^*, \varphi_T^*) \leq (T+2)^m \cdot \sum_{\mathbf{x}} \sum_{\mathbf{X}} p(\mathbf{x}|\mathbf{X}) P(\mathbf{X}) \cdot \tag{2.10}$$

$$\cdot \sum_{t=T+1}^{L} M_t(\mathbf{X}|\mathbf{x}) \cdot \left(\frac{2}{q} + \beta_t \right)^m,$$

where

$$\beta_t = \begin{cases} \binom{t}{t/2} \cdot 2^{-t}, & \text{if } t \text{ is even}, \\ \binom{t}{(t-1)/2} \cdot 2^{-t}, & \text{if } t \text{ is odd}, \end{cases} \tag{2.11}$$

and m is the largest integer such that

$$m \leq \frac{l}{\log_2 q}. \tag{2.12}$$

Corollary. Let

$$T + 2 = (2\pi\varepsilon L)^{\tau/2} \tag{2.13}$$

and

$$P(\mathbf{X}) > 0 \quad \Rightarrow \quad \mathbf{x}_i \neq \mathbf{x}_j \quad \text{for all } i \neq j, \tag{2.14}$$

$$M = \sum_{t=0}^{\varepsilon L} \left(\begin{array}{c} L \\ t \end{array} \right). \tag{2.15}$$

Then

$$N_T(p, P) \leq M \cdot 2^{-l(1-\tau)} \cdot 2^{L\delta_{L,l}(\varepsilon)}, \tag{2.16}$$

where

$$\delta_{L,l}(\varepsilon) = \frac{\log_2 \varepsilon L}{L} + \frac{l}{L} \cdot \frac{8}{\log_2 2\pi\varepsilon L}. \tag{2.17}$$

The proofs are given in Section 5.

Discussion.

1. The key point of the statement above is the fact that the expression at the right hand side of (2.16) does not depend on p and P. Let us suppose that $T = 0$ and the pattern is one of the records of the DB, chosen in accordance with the uniform distribution. Suppose also that the first l bits of each record are stored in RAM. If all the records are random vectors whose components are independently chosen from $\{0, 1\}$ with the probability $1/2$, then the average number of 'extra' transmissions to RAM is equal to $M \cdot 2^{-l}$, and we cannot improve this result. Let us change the input data in the following way : suppose that $\mathbf{x}_1$ is chosen as before, but all the other records are different random vectors, chosen from the set which consists of the vectors $\mathbf{x}'$ such that $d_H(\mathbf{x}_1, \mathbf{x}') \leq \varepsilon L$, where the parameter ε is defined in (2.15); in computer science, such a construction is called a cluster [4]. Then the average number of 'extra' transmissions to RAM is approximately equal to $M \cdot 2^{-lH(\varepsilon)}$, where $H(\varepsilon) = -\varepsilon \log_2 \varepsilon - (1 - \varepsilon) \log_2(1 - \varepsilon)$ is the binary entropy function ($M \approx 2^{LH(\varepsilon)}$, 'almost all' records and 'almost all' patterns are located

at distance εL from $\mathbf{x}_1$, and there are approximately $2^{lH(\varepsilon)}$ groups of records of size $2^{(L-l)H(\varepsilon)}$ having the same prefixes of length l). Thus, the characteristics of the hashing algorithm can be essentially different in the first and in the second cases. Furthermore, it is not evident how to organize a hashing procedure in the second case to get $M \cdot 2^{-l}$ as a number of 'extra' transmissions, since the position of the cluster is a uniformly distributed random variable. The theorem above shows that such a procedure exists, and the proof gives a construction of the hash function.

2. The proof of the Theorem shows that in order to calculate $f^*(\mathbf{x}_j)$ we find the Hamming distances between $\mathbf{x}_j$ and $l/\log_2 q_0$ binary vectors that are chosen in a special way and then find the Lee distances between these numbers and $l/\log_2 q_0$ integers stored in RAM; $q_0 = (2\pi\varepsilon L)^{1/2}$. To calculate $\varphi_T^*(\mathbf{x}, f^*(\mathbf{x}_j))$, we compare the obtained Lee distances with the threshold value T. The complexity of these operations and the required size of RAM may be neglected in compare to the computational efforts needed to access the external memory and transmit a record to RAM.

3. A simple generalization of the proof of the Theorem allows us to claim that there exists the same pair (f^*, φ_T^*), providing the bound (2.10) multiplied by $L^{\tau/2}$, for all $T < L^{\tau/2}$. This increment practically does not affect the exponent of the bound when L is large enough.

3 Estimations of the Hamming Distance Between Binary Vectors Based on Indirect Observations

Let $\mathbf{x}, \mathbf{x}_j$ and $\mathbf{c}$ be some binary vectors of length L. Then, due to the triangle inequality for the Hamming distances, we write :

$$d_H(\mathbf{x}, \mathbf{x}_j) \geq |d_H(\mathbf{c}, \mathbf{x}) - d_H(\mathbf{c}, \mathbf{x}_j)| \tag{3.1}$$

and

$$|d_H(\mathbf{c}, \mathbf{x}) - d_H(\mathbf{c}, \mathbf{x}_j)| > T \quad \Rightarrow \quad d_H(\mathbf{x}, \mathbf{x}_j) > T. \tag{3.2}$$

Hence, we can select a binary vector $\mathbf{c}$ and assign the encoding $f(\mathbf{x}_j)$ as the binary representation of $d_H(\mathbf{c},\mathbf{x}_j)$, and the decoding as

$$\varphi_T(\mathbf{x}, f(\mathbf{x}_j)) = \chi\{\ j : |d_H(\mathbf{c},\mathbf{x}) - d_H(\mathbf{c},\mathbf{x}_j)| \le T\ \}.$$

This hashing method is illustrated in Appendix 2.

If $\mathbf{c} \in \{0,1\}^L$ is chosen at random in such a way that every component takes values 0 and 1 independently of other components with probability 1/2, then

$$Pr\{\ d_H(\mathbf{c},\mathbf{x}_j) = d_j\ \mid\ \mathbf{x}_j\ \} = B_L(d_j), \tag{3.3}$$

where

$$B_L(d_j) = \begin{pmatrix} L \\ d_j \end{pmatrix} \cdot 2^{-L} \text{ for all } d_j = 0, ..., L \tag{3.4}$$

is the binomial distribution. It is well-known [8], that the entropy of this distribution is equal to

$$H(B_L) = -\sum_{d=0}^{L} B_L(d) \log_2 B_L(d) \simeq -\log_2 \sqrt{\pi L/2}.$$

This means that if we observe a sequence of m independent random variables, such that every component is distributed in accordance with the binomial distribution and m is large enough, then, roughly speaking, it is sufficient to have approximately $m \cdot \log_2 \sqrt{\pi L/2}$ bits to store this sequence. In our case, we can, in principle, get this sequence, selecting $m \le l/\log_2(L+1)$ vectors $\mathbf{c}_1, ..., \mathbf{c}_m$ at random, and assign the encoding of $\mathbf{x}_j$ as the binary representation of $d_H(\mathbf{c}_1,\mathbf{x}_j), ..., d_H(\mathbf{c}_m,\mathbf{x}_j)$, but m cannot be large compared to L. Besides, we need to provide the property that any record of the DB, which is close to a given pattern, should not be missed in the first step of the decoding procedure.

Discussion.

1. If $T = 0$, then any function f, transforming binary vectors of length L to binary vectors of length l may be used for encoding. Then the decoding consists of calculation of the value of this function for a given pattern and selection of all the records of the DB, which have the same value. However, if $T > 0$, then using a function, which is a distance,

i.e., the triangle inequality is valid, we can decrease the computational efforts compared to an arbitrary function. The distance function, as it follows from our results, can be also effectivelly used when $T = 0$.

2. The triangle inequality is known as a tool, which allows us essentially reduce the number of measurements of the Hamming distances in the problem of finding the neibouring points in random digital fields [7]. This problem can be represented as follows. Suppose, we are given M binary vectors, $\mathbf{x}_1, ..., \mathbf{x}_M$, of length L and we want to find all the pairs (i, j) such that $d_H(\mathbf{x}_i, \mathbf{x}_j) \leq r$, where r is a given constant. The direct solution of this problem requires $M(M-1)/2$ measurements. However, as it was shown in [7], we can fix $m = O(\ln M / \ln \sqrt{L})$ and measure mM distances $d_H(\mathbf{x}_i, \mathbf{x}_k)$, $i = 1, ..., M$, $k = 1, ..., m$, i.e., we fill in the first m columns and the first m rows of the matrix consisting of the Hamming distances. Using the triangle inequality, we can exclude a number of positions of that matrix, since we definitely know that the corresponding vectors are at distance which is greater than r. The number of positions, where we cannot come to such a conclusion, is $O(M \ln M)$. These considerations are valid if $r \leq \sqrt{2\alpha L}$, where α is defined by the equation $1 - H(\alpha) = \log_2 M/L$, and when we are interested in the average characteristics taken over the ensemble of independent and uniformly distributed vectors $\mathbf{x}_1, ..., \mathbf{x}_M$. Our considerations can be presented as an extension of the approach of [7] in two direction : 1) we are also interested in the uniform distribution on integers, which are the elements of the first m columns and the first m rows of the matrix of distances; 2) we are dealing with any probability distributions on the collections of the vectors $\mathbf{x}_1, ..., \mathbf{x}_M$.

The idea of introducing the Lee distance in our problem is based on an observation that, mainly, the pairs of binary vectors of length L are located at the Hamming distance, which is close to $L/2$. Therefore, we expect, that direct use of (3.1) to restrict the domain of the search leads to a non-optimal tradeoff between the number of 'extra' transmissions of records from the external memory and the size of RAM, required to store the results of preprocessing.

Let q be an integer, and $d_H^*(\mathbf{c}, \mathbf{x}_j)$ denote the remainder of division of $d_H(\mathbf{c}, \mathbf{x}_j)$ by q. Then

$$0 \leq d_H^*(\mathbf{c}, \mathbf{x}_j) \leq q - 1$$

and

$$d_H(\mathbf{c},\mathbf{x}_j) = d_H^*(\mathbf{c},\mathbf{x}_j) + sq \ \text{ for some } \ s \in \{0,1,...\}.$$

Therefore,

$$|d_H(\mathbf{c},\mathbf{x}) - d_H(\mathbf{c},\mathbf{x}_j)| \geq d_{Lee}(\ d_H(\mathbf{c},\mathbf{x}), d_H^*(\mathbf{c},\mathbf{x}_j) \ ), \tag{3.5}$$

where

$$d_{Lee}(\ d_H(\mathbf{c},\mathbf{x}), d_H^*(\mathbf{c},\mathbf{x}_j) \ ) = \min_{s=0,1,...} |d_H(\mathbf{c},\mathbf{x}) - (d_H^*(\mathbf{c},\mathbf{x}_j) + sq)| \tag{3.6}$$

is the Lee distance between $d_H(\mathbf{c},\mathbf{x})$ and $d_H^*(\mathbf{c},\mathbf{x}_j)$. However, in accordance with (3.1), the expression at the left hand side of (3.5) is a lower bound on $d_H(\mathbf{x},\mathbf{x}_j)$. Thus,

$$d_{Lee}(\ d_H(\mathbf{c},\mathbf{x}), d_H^*(\mathbf{c},\mathbf{x}_j) \ ) > T \quad \Rightarrow \quad d_H(\mathbf{x},\mathbf{x}_j) > T. \tag{3.7}$$

It means, that we can assign an encoding and decoding, satisfying (2.2) and (2.3), based on the Lee distance. To store a value of the Lee distance, we need to have only $\log_2 q$ bits instead of $\log_2(L+1)$ bits for a value of the Hamming distance, and this hashing method may lead to more effective tradeoff between the accuracy of a lower bound on $d_H(\mathbf{x},\mathbf{x}_j)$ and the required size of RAM. We will examine this point in the next sections, and the properties of the Lee distance, which are given below, will be used.

Lemma 1. If q is even, then the following inequality :

$$Pr\{ \ d_H^*(\mathbf{c},\mathbf{x}_j) = d_j^* \ \mid \ \mathbf{x}_j \ \} \leq \frac{1}{q} + \beta_L \tag{3.8}$$

is valid for any $d \geq 0$, and any $\mathbf{x}_j \in \{0,1\}^L$, where β_L is defined in (2.11).

Lemma 2. Let $\mathbf{x}, \mathbf{x}_j \in \{0,1\}^L$ be chosen in such a way that

$$d_H(\mathbf{x},\mathbf{x}_j) = t,$$

q is even and $k < q/2$. If both t and k are even or odd, then

$$Pr\{ \ d_{Lee}(d_H(\mathbf{c},\mathbf{x}), d_H^*(\mathbf{c},\mathbf{x}_j)) = k \ \mid \ \mathbf{x},\mathbf{x}_j \ \} \leq \frac{4}{q} + 2\beta_t. \tag{3.9}$$

Otherwise,

$$Pr\{ \ d_{Lee}(d_H(\mathbf{c},\mathbf{x}), d_H^*(\mathbf{c},\mathbf{x}_j)) = k \ \mid \ \mathbf{x},\mathbf{x}_j \ \} = 0.$$

The proofs of Lemmas 1, 2 are given in Appendix 1.

Discussion. Let $\mathbf{x}_j$ be given and $\mathbf{c}$ be chosen at random in accordance with the uniform distribution. Let us calculate the Hamming distance $d = d_H(\mathbf{c}, \mathbf{x}_j)$ and generate a new random variable d^*, which is equal to 0 if d is even and equal to 1 if d is odd, i.e., $d^* = d$, mod 2. It is well-known that if L is odd, then

$$\sum_{d-\text{even}} B_L(d) = \sum_{d-\text{odd}} B_L(d) = 1/2.$$

The sums in this equation are the probabilities that $d^* = 0$ and $d^* = 1$. Thus, we obtain a uniformly distributed bit. To get a good characteristics of the hashing algorithm, we need to keep this property and reduce d modulo $q > 2$, since we want to use the triangle inequality and estimate the Hamming distance between $\mathbf{x}_j$ and $\mathbf{x}$ when $T > 0$. The results of Lemmas 1,2 show that it is possible if

$$q < \beta_L^{-1} \simeq \sqrt{\pi L/2}.$$

Note also that the probability, written at the left hand side of (3.8), will be obtained if we pick up every q-th term of the binomial distribution $B_L(d_j)$ when the first term is equal to d_j^*. This procedure is illustrated in Fig.1.

4 A Hashing Algorithm

Let us assign an even q and an integer m, satisfying (2.12). Besides, let us assign binary vectors $\mathbf{c}_1, ..., \mathbf{c}_m \in \{0,1\}^L$ and assume that they are stored in RAM. Let $d_H^*(\mathbf{c}_i, \mathbf{x}_j)$ be the Hamming distance between $\mathbf{c}_i$ and $\mathbf{x}_j$, reduced modulo q, and let

$$\begin{aligned} f^*(\mathbf{x}_j) &= (d_H^*(\mathbf{c}_1, \mathbf{x}_j), ... d_H^*(\mathbf{c}_m, \mathbf{x}_j)), \qquad (4.1) \\ \varphi_T^*(\mathbf{x}, f(\mathbf{x}_j)) &= \prod_{i=1}^{m} \chi\{ \; d_{Lee}(\; d_H(\mathbf{c}_i, \mathbf{x}), d_H^*(\mathbf{c}_i, \mathbf{x}_j) \le T \;) \; \}. \end{aligned}$$

Using (3.7), we conclude that this pair (encoding, decoding) satisfies (2.2) and (2.3). The hashing algorithm is illustrated in Appendix 2.

5 Proof of the Theorem and Corollary

Let (f^*, φ_T^*) be a pair (encoding, decoding) defined in (4.1), which leads to the minimal average number of 'extra' transmissions, where the minimum is taken on all binary vectors $\mathbf{c}_1, ..., \mathbf{c}_m \in \{0,1\}^L$, i.e.,

$$\begin{aligned} N_T(p, P|f^*, \varphi_T^*) \;\; = \;\; & \min_{\mathbf{c}_1,...,\mathbf{c}_m} \sum_{\mathbf{x}} \sum_{\mathbf{X}} p(\mathbf{x}|\mathbf{X}) P(\mathbf{X}) \cdot \\ & \cdot \sum_{j=1}^{M} \chi\{ \;\; d_H(\mathbf{x}, \mathbf{x}_j) > T \;\; \} \cdot \\ & \cdot \prod_{i=1}^{m} \chi\{ \;\; d_{Lee}(\;\; d_H(\mathbf{c}_i, \mathbf{x}), d_H^*(\mathbf{c}_i, \mathbf{x}_j) \;\;) \le T \;\; \}. \end{aligned} \tag{5.1}$$

Let us introduce the code ensemble, consisting of all collections of m binary vectors of length L, such that each component of each vector is an independent binary variable, equaled 0 or 1 with probability 1/2. Then we can use the random coding arguments [8] and claim that the minimal value of the expression at the right hand side of (5.1), taken on all possible vectors $\mathbf{c}_1, ..., \mathbf{c}_m$, is not greater than the value obtained in average over the code ensemble. Hence,

$$\begin{aligned} N_T(p, P|f^*, \varphi_T^*) \;\; \le \;\; & \sum_{\mathbf{x}} \sum_{\mathbf{X}} p(\mathbf{x}|\mathbf{X}) P(\mathbf{X}) \cdot \sum_{j=1}^{M} \chi\{ \;\; d_H(\mathbf{x}, \mathbf{x}_j) > T \;\; \} \cdot \\ & \cdot \prod_{i=1}^{m} \sum_{\mathbf{c}_i} 2^{-L} \cdot \chi\{ \;\; d_{Lee}(\;\; d_H(\mathbf{c}_i, \mathbf{x}), d_H^*(\mathbf{c}_i, \mathbf{x}_j) \;\;) \le T \;\; \} \\ = \;\; & \sum_{\mathbf{x}} \sum_{\mathbf{X}} p(\mathbf{x}|\mathbf{X}) P(\mathbf{X}) \cdot \sum_{j=1}^{M} \sum_{t=T+1}^{L} \chi\{ \;\; d_H(\mathbf{x}, \mathbf{x}_j) = t \;\; \} \cdot \\ & \cdot \left(\sum_{k=0}^{T} Pr\{ \;\; d_{Lee}(d_H(\mathbf{c}, \mathbf{x}), d_H^*(\mathbf{c}, \mathbf{x}_j)) = k \;\; | \;\; \mathbf{x}, \mathbf{x}_j \;\; \} \right)^m . \end{aligned}$$

Therefore, the result of Lemma 2 leads to the upper bound :

$$\begin{aligned} N_T(p, P|f^*, \varphi_T^*) \;\; \le \;\; & \sum_{\mathbf{x}} \sum_{\mathbf{X}} p(\mathbf{x}|\mathbf{X}) P(\mathbf{X}) \cdot \sum_{j=1}^{M} \sum_{t=T+1}^{L} \chi\{ \;\; d_H(\mathbf{x}, \mathbf{x}_j) = t \;\; \} \cdot \\ & \cdot \sum_{k=0}^{T} \left(\frac{4}{q} + 2\beta_t \right)^m \cdot \chi\{ \;\; \text{both } k \text{ and } t \text{ are even or odd} \;\; \} \end{aligned}$$

$$\leq \sum_{\mathbf{x}} \sum_{\mathbf{X}} p(\mathbf{x}) P(\mathbf{X}) \cdot \sum_{t=T+1}^{L} M_t(\mathbf{X}|\mathbf{x}) \cdot \frac{(T+2)^m}{2^m} \cdot \left(\frac{4}{q} + 2\beta_t\right)^m .$$

This inequality coincides with (2.10), and Theorem has been proved.

To prove the Corollary we minimize the expression at the right hand side of (2.10) on all integers $M_t(\mathbf{X}|\mathbf{x})$, satisfying the restrictions :

$$\begin{aligned} M_t(\mathbf{X}|\mathbf{x}) &\leq \binom{L}{t}, \\ \sum_{t=0}^{L} M_t(\mathbf{X}|\mathbf{x}) &= M. \end{aligned}$$

Hence, if M satisfies (2.15), then the numbers $M_t(\mathbf{X}|\mathbf{x})$ should be as large as possible for $t = 0, ..., \varepsilon L$ and all the other numbers should be equal to zero. Therefore,

$$\begin{aligned} N_T(p, P) &\leq (T+2)^m \cdot \sum_{t=T+1}^{\varepsilon L} \binom{L}{t} \left(\frac{2}{q} + \beta_t\right)^m \\ &\leq M \cdot (T+2)^m \cdot \varepsilon L \cdot \left(\frac{2}{q} + \beta_{\varepsilon L}\right)^m . \end{aligned} \tag{5.2}$$

Let us assign $q = \sqrt{2\pi\varepsilon L}$. Then $2/q \geq \beta_{\varepsilon L}$ and using (2.12), (2.13), and (5.2) we write

$$\begin{aligned} N_T(p, P) &\leq M \cdot \varepsilon L \cdot exp_2 \left\{ \frac{l}{\log_2 q} \cdot \left(\log_2(T+2) + \log_2 \frac{2}{q} \right) \right\} \\ &= M \cdot \varepsilon L \cdot exp_2 \left\{ l \cdot \left(\frac{\log_2(T+2)}{\log_2 q} + \frac{2}{\log_2 q} - 1 \right) \right\} \\ &= M \cdot \varepsilon L \cdot exp_2 \left\{ l \cdot \left(\tau + \frac{4}{\log_2 q} - 1 \right) \right\}, \end{aligned} \tag{5.3}$$

where $exp_2 z = 2^z$ for all z. Using (2.17) we conclude that the expression at the right hand side of (5.3) coincides with (2.16).

6 An Example of Hashing Procedure

Let us suppose that the DB consists of the binary representations (in ASCII codes) of the names of 50 states of U.S.A., written in capital letters. Then $M = 50$ and $L = 13 \cdot 8 = 104$, since 13 is the maximal length of the names (for example, MASSACHUSETTS) and we assume that the gaps are added to each name of length less than 13.

This example was considered in [10], where a hashing procedure, which assigns a unique integer, belonging to the set $\{1, ..., 53\}$, to each name, was proposed. The procedure is based on the observation that it is possible to pick up two letters, k_1 and k_2, from each name in such a way that all the pairs (k_1, k_2) are different. An integer, corresponding to the pair (k_1, k_2), is assigned as $v_1(k_1) + v_2(k_2)$, where v_1 and v_2 are integer-valued functions constructed in a special way. The letters k_1 and k_2 were chosen from a name of length r in accordance with the following algorithm :

1) if $r = 9$, then k_1 is the last letter and k_2 is the third letter;

2) if $r \neq 9$, then k_1 is the first letter and k_2 is the $(7 - \lfloor r/2 \rfloor)$-th letter, where $\lfloor r/2 \rfloor$ denotes the maximal integer such that $\lfloor r/2 \rfloor \leq r/2$.

A generalization of this procedure to the other DBs seems to be difficult.

Let us suppose that $l = 8$, i.e., one byte can be used to store a representative of each name in RAM. If we assign two codewords of length 104, then this byte can be constructed in such a way that the $4k$-th,..., the $4k+3$-rd bits contain the Hamming distance between the element of the DB and the k-th codeword, reduced modulo 16, where $k = 0, 1$. For example, the codewords can be given by the matrix (we use the hexadecimal notations)

$$\mathbf{C} = \left\| \begin{array}{ccccccccccccc} \mathrm{A1} & \mathrm{C6} & 87 & \mathrm{B4} & \mathrm{DD} & 52 & 23 & 20 & \mathrm{D9} & \mathrm{9E} & \mathrm{7F} & \mathrm{4C} & 95 \\ \mathrm{AA} & \mathrm{9B} & 38 & 11 & 76 & 77 & \mathrm{E4} & \mathrm{4D} & 02 & 13 & 50 & 49 & \mathrm{4E} \end{array} \right\|.$$

Then we obtain different bytes for 42 names and 3 groups of names having the same byte. These groups are as follows :

{ *ALASKA*, *MONTANA*, *NEWYORK* },
{ *HAWAII*, *INDIANA*, *PENNSYLVANIA* }.
{ *ALABAMA*, *TENNESSEE* }.

If we assume that each name appears as a pattern with the probability 1/50 and we want to find this name in the DB (the case $T = 0$), then the average number of 'extra' transmissions to RAM is equal to

$$2 \cdot 3/50 + 2 \cdot 3/50 + 1 \cdot 2/50 = 14/50.$$

Since 211 bytes are not the values of our hash function, we can use the triangle inequality to correct some misprints in the patterns (the case $T > 0$). Note that we also need a 'good' code, which assigns a byte to each letter, to realize this possibility in an effective way.

The matrix $\mathbf{C}$, given above, is a result of the simulation, where we examined different matrices, chosen at random, and selected the matrices that provide a minimal average number of 'extra' transmissions. The estimate of the average value of this number, taken over the matrices we have examined, is equal to 0.60, while the estimate of the square root of the variation is equal to 0.15. We assume that if the size of the DB is large enough, then the average number of 'extra' transmissions is a very stable random variable in the code ensemble, and some analytical upper bounds on the moments of this variable confirm the assumption. More detailed discussion of this point will be proposed in the future.

7 Acknowledgement

The author is much obliged to Prof. Valery Koshelev for the support and encouragement. The author is also grateful to Prof. Rudolf Ahlswede and Dr. Göran Lindh for interesting discussions, and anonymous reviewers for helpfull comments.

Appendix 1

Proof of Lemma 1

Suppose that L is even, and denote the elements of the residue class $[d_j^*]$ by $d^{(s)}, s = 0, \pm 1, ...,$ in such a way that (Fig.2)

$$\begin{array}{rcl} d^{(0)} & \in & [\ L/2 - q/2,\ \ L/2 + q/2\ ), \\ d^{(s)} & = & d^{(0)} + sq \quad \text{for all} \ \ s = \pm 1, \pm 2, ... \end{array}$$

Then we can write

$$Pr\{\ d_H^*(\mathbf{c}, \mathbf{x}_j) = d_j^* \ \mid \ \mathbf{x}_j \ \} = \sum_s B_L(d^{(s)}). \tag{A.1}$$

Suppose that $d^{(0)} \geq L/2$. Then

$$\begin{array}{rcl} B_L(L/2 + sq) & \leq \ B_L(d^{(s)}) \ \leq & B_L(L/2 + q/2 + sq) \\ & & \text{for all} \ \ s = -1, -2, ..., \\ B_L(L/2 + q/2 + sq) & \leq \ B_L(d^{(s)}) \ \leq & B_L(L/2 + sq) \\ & & \text{for all} \ \ s = 0, 1, 2, ... \end{array} \tag{A.2}$$

Using (A.1) and (A.2), we conclude that

$$b \leq Pr\{\ d_H^*(\mathbf{c}, \mathbf{x}_j) = d_j^* \ \mid \ \mathbf{x}_j \ \} \leq b + B_L(L/2), \tag{A.3}$$

where

$$b = \sum_{s>0} B_L(L/2 + sq/2).$$

Inequalities (A.3) follow from the identities :

$$\begin{array}{rcl} B_L(L/2 + sq/2) & = & B_L(L/2 - sq/2), \\ B_L(L/2 + q/2 + sq) & = & B_L(L/2 - q/2 - sq) \ \text{for all} \ \ s = -1, -2, ... \end{array}$$

It is easy to see that inequalities (A.3) are also valid for $d^{(0)} < L/2$. Furthermore, they are valid for all $d_j^* \geq 0$. Since

$$\sum_{d_j^*=0}^{q-1} Pr\{\ d_H^*(\mathbf{c}, \mathbf{x}_j) = d_j^* \ \mid \ \mathbf{x}_j \ \} = 1,$$

using the first inequalities in (A.3), we obtain :

$$b \leq \frac{1}{q}.$$

Substitution of this estimate to the second inequality in (A.3) completes the proof for even L. The proof for odd L is similar.

Proof of Lemma 2

Let I be the set that consists of all indices, for which $\mathbf{x}$ differs from $\mathbf{x}_j$, and let

$$I^{'} = \{1, ..., L\} \backslash I.$$

Let us divide the vector $\mathbf{x}$ into two vectors : $\mathbf{x}_I$ and $\mathbf{x}_{I'}$ in such a way that $\mathbf{x}_I$ consists of all components $x_i, i \in I$, and $\mathbf{x}_{I'}$ consists of all components $x_i, i \in I^{'}$. Besides, let us assign a vector $\mathbf{c} \in \{0,1\}^L$ and divide it into two vectors $\mathbf{c}_I$ and $\mathbf{c}_{I'}$ in a similar way. If

$$d_H(\mathbf{c}_I, \mathbf{x}_I) = d,$$

then

$$d_H(\mathbf{c}, \mathbf{x}) = d + d_H(\mathbf{c}_{I'}, \mathbf{x}_{I'}).$$

Since the Hamming distance between $\mathbf{x}$ and $\mathbf{x}_j$ is equal to t, we can write :

$$d_H(\mathbf{c}, \mathbf{x}_j) = t - d + d_H(\mathbf{c}_{I'}, \mathbf{x}_{I'}).$$

Therefore, using (3.6) we obtain :

$$\begin{aligned} & d_{Lee}(\ \ d_H(\mathbf{c}, \mathbf{x}), d^*_H(\mathbf{c}, \mathbf{x}_j)\ \ ) = \\ = \ & d_{Lee}(\ \ d + d_H(\mathbf{c}_{I'}, \mathbf{x}_{I'}), t - d + d_H(\mathbf{c}_{I'}, \mathbf{x}_{I'})\ \ ) = \\ = \ & d_{Lee}(\ \ d, t - d\ \ ). \end{aligned}$$

Thus,

$$\begin{aligned} & Pr\{\ \ d_{Lee}(d_H(\mathbf{c}, \mathbf{x}), d^*_H(\mathbf{c}, \mathbf{x}_j)) = k\ \ |\ \ \mathbf{x}, \mathbf{x}_j\ \ \} = \\ = & \sum_{\mathbf{c}_I \in \{0,1\}^t} 2^{-t} \cdot \sum_{d=0}^{t} \chi\{\ \ d_H(\mathbf{c}_I, \mathbf{x}_I) = d\ \ \} \cdot \chi\{\ \ d_{Lee}(d, t-d) = k\ \ \}. \end{aligned}$$

If

$$d_{Lee}(\ d, t-d\ ) = k,$$

then there exists an integer s such that

$$|d-(t-d+sq)| = k,$$

and either

$$2d = t-k+sq \tag{A.4}$$

or

$$2d = t+k+sq. \tag{A.5}$$

Inequalities (A.4) and (A.5) may be satisfied iff both t and k are either even or odd. Then

$$\begin{aligned} & Pr\{\ d_{Lee}(d_H(\mathbf{c},\mathbf{x}), d_H^*(\mathbf{c},\mathbf{x}_j)) = k \ \mid\ \mathbf{x},\mathbf{x}_j\ \} \\ = \ & Pr\{\ d_H(\mathbf{c}_I,\mathbf{x}_I) \in [(t-k)/2]_{q/2} \ \mid\ \mathbf{x}_I\ \} \\ + \ & Pr\{\ d_H(\mathbf{c}_I,\mathbf{x}_I) \in [(t+k)/2]_{q/2} \ \mid\ \mathbf{x}_I\ \}, \end{aligned}$$

where $[i]_{q/2}$ denotes the residue class modulo $q/2$, i.e., $[i]_{q/2}$ consists of integers $i^* + sq/2, s = 0, 1, ...,$ where $i^* \in [0, q/2)$ is assigned in such a way that $i - i^* = sq/2$ for some integer s. Therefore, substitution of the leaders of the residue classes $[(t+k)/2]_{q/2}$ and $[(t-k)/2]_{q/2}$ for d_j^*, t for L, and q for $q/2$ to the result of Lemma 1 completes the proof.

Appendix 2
Illustrations of Hashing Methods

Let
$$M = 4, \quad L = 8, \quad l = 4,$$
and
$$\mathbf{X} = \left\| \begin{array}{c} \mathbf{x}_1 \\ \cdot \\ \cdot \\ \mathbf{x}_4 \end{array} \right\| = \left\| \begin{array}{cccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right\|.$$
Let us note that $l = 4$ bits of RAM may be used for storing the Hamming distance between any two binary vectors of length $L = 8$. Suppose that
$$\mathbf{C} = \left\| \; \mathbf{c}_1 \; \right\| = \left\| \; 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \; \right\|.$$
Then
$$\left\| \begin{array}{c} d_H(\mathbf{c}_1, \mathbf{x}_1) \\ \cdot \\ \cdot \\ d_H(\mathbf{c}_1, \mathbf{x}_4) \end{array} \right\| = \left\| \begin{array}{c} 0 \\ 5 \\ 3 \\ 1 \end{array} \right\|.$$
The values of the hash function can be defined as binary representations of these integers :
$$\left\| \begin{array}{c} f(\mathbf{x}_1) \\ \cdot \\ \cdot \\ f(\mathbf{x}_4) \end{array} \right\| = \left\| \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{array} \right\|.$$
Let
$$\mathbf{x} = (0, 0, 0, 0, 0, 0, 0, 1).$$
The vector x differs from all rows of the matrix $\mathbf{X}$ and does not coincide with the row $\mathbf{x}_1$ in one bit. Therefore, using (2.1) we write :
$$J_0(\mathbf{x}, \mathbf{X}) = \emptyset, \quad J_1(\mathbf{x}, \mathbf{X}) = \{1\}.$$
Since $d_H(\mathbf{c}_1, \mathbf{x}) = 1$, to select the rows, which can coincide with $\mathbf{x}$, we find the indices j such that
$$d_H(\mathbf{c}_1, \mathbf{x}_j) = 1,$$

and to select the rows, which can differ from $\mathbf{x}$ in one bit, we find the indices j such that

$$d_H(\mathbf{c}_1, \mathbf{x}_j) \in \{0, 1, 2\}.$$

Then we will construct the sets :

$$\begin{array}{rcl} J_0(\mathbf{x}, \mathbf{X}|f, \varphi_0) & = & \{4\}, \\ J_1(\mathbf{x}, \mathbf{X}|f, \varphi_1) & = & \{1, 4\} \end{array}$$

and

$$\begin{array}{rcl} N_0(\mathbf{x}, \mathbf{X}|f, \varphi_0) & = & 1 - 0 = 1, \\ N_1(\mathbf{x}, \mathbf{X}|f, \varphi_1) & = & 2 - 1 = 1. \end{array}$$

The hashing method, that includes the previous one as a special case, consists in the assignment of integers m and q, connected by (2.12), assignment of the matrix $\mathbf{C}$ of size $m \times L$, and calculation of $d_H^*(\mathbf{c}_i, \mathbf{x}_j)$, where $i = 1, ..., m, j = 1, ..., M$. Let

$$m = 2, \quad q = 4,$$

and

$$\mathbf{C} = \left\| \begin{array}{c} \mathbf{c}_1 \\ \mathbf{c}_2 \end{array} \right\| = \left\| \begin{array}{cccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{array} \right\|.$$

Then

$$\left\| \begin{array}{c} (d_H(\mathbf{c}_1, \mathbf{x}_1), d_H(\mathbf{c}_2, \mathbf{x}_1)) \\ \cdot \\ \cdot \\ (d_H(\mathbf{c}_1, \mathbf{x}_4), d_H(\mathbf{c}_2, \mathbf{x}_4)) \end{array} \right\| = \left\| \begin{array}{c} (0, 5) \\ (5, 2) \\ (3, 6) \\ (1, 6) \end{array} \right\|$$

and

$$\left\| \begin{array}{c} (d_H^*(\mathbf{c}_1, \mathbf{x}_1), d_H^*(\mathbf{c}_2, \mathbf{x}_1)) \\ \cdot \\ \cdot \\ (d_H^*(\mathbf{c}_1, \mathbf{x}_4), d_H^*(\mathbf{c}_2, \mathbf{x}_4)) \end{array} \right\| = \left\| \begin{array}{c} (0, 1) \\ (1, 2) \\ (3, 2) \\ (1, 2) \end{array} \right\|.$$

The values of the hash-function are binary representations of these integers so that two bits are used to store each integer :

$$\left\| \begin{array}{l} f(\mathbf{x}_1) \\ \cdot \\ \cdot \\ f(\mathbf{x}_4) \end{array} \right\| = \left\| \begin{array}{cccc} 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{array} \right\|.$$

Then

$$
\begin{aligned}
(d_H(\mathbf{c}_1, \mathbf{x}), d_H(\mathbf{c}_2, \mathbf{x})) &= (1, 4), \\
(d_H^*(\mathbf{c}_1, \mathbf{x}), d_H^*(\mathbf{c}_2, \mathbf{x})) &= (1, 0),
\end{aligned}
$$

and to construct the set $J_0(\mathbf{x}, \mathbf{X})$ we select the indices j such that

$$
\begin{aligned}
d_H^*(\mathbf{c}_1, \mathbf{x}_j) &= 1, \\
d_H^*(\mathbf{c}_2, \mathbf{x}_j) &= 0,
\end{aligned}
$$

and to construct the set $J_1(\mathbf{x}, \mathbf{X})$ we select the indices j such that

$$
\begin{array}{lclllll}
d_H^*(\mathbf{c}_1, \mathbf{x}_j) & \in & \{ & 1 - 1 = 0, & 1, & 1 + 1 = 2 & \}, \\
d_H^*(\mathbf{c}_2, \mathbf{x}_j) & \in & \{ & 0 - 1 + 4 = 3, & 0, & 0 + 1 = 1 & \}.
\end{array}
$$

It leads to the following equations :

$$
\begin{array}{lclcl}
J_0(\mathbf{x}, \mathbf{X} | f_{\mathbf{C}}, \varphi_0) & = & \{2, 4\} \bigcap \emptyset & = & \emptyset, \\
J_1(\mathbf{x}, \mathbf{X} | f_{\mathbf{C}}, \varphi_1) & = & \{1, 2, 4\} \bigcap \{1\} & = & \{1\}
\end{array}
$$

and

$$
\begin{aligned}
N_0(\mathbf{x}, \mathbf{X} | f, \varphi_0) &= 0, \\
N_1(\mathbf{x}, \mathbf{X} | f, \varphi_1) &= 0.
\end{aligned}
$$

Note, that the last hashing algorithm leads to the assignment of the same values of the hash-function to the rows $\mathbf{x}_2$ and $\mathbf{x}_4$, while the Hamming distance between these rows is equal to 6.

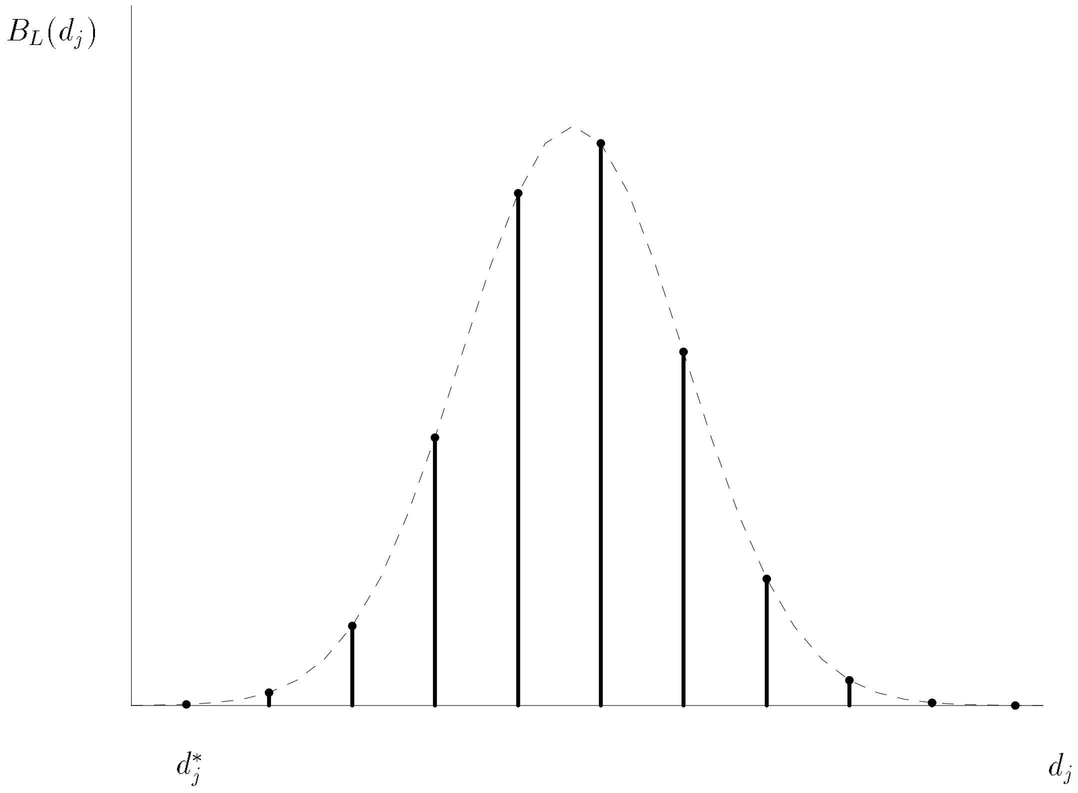

Figure 1: The probability at the right hand side of (3.8) is equal to the sum of ordinates of the impulses; $d_j = d_j^*$ defines the first impulse.

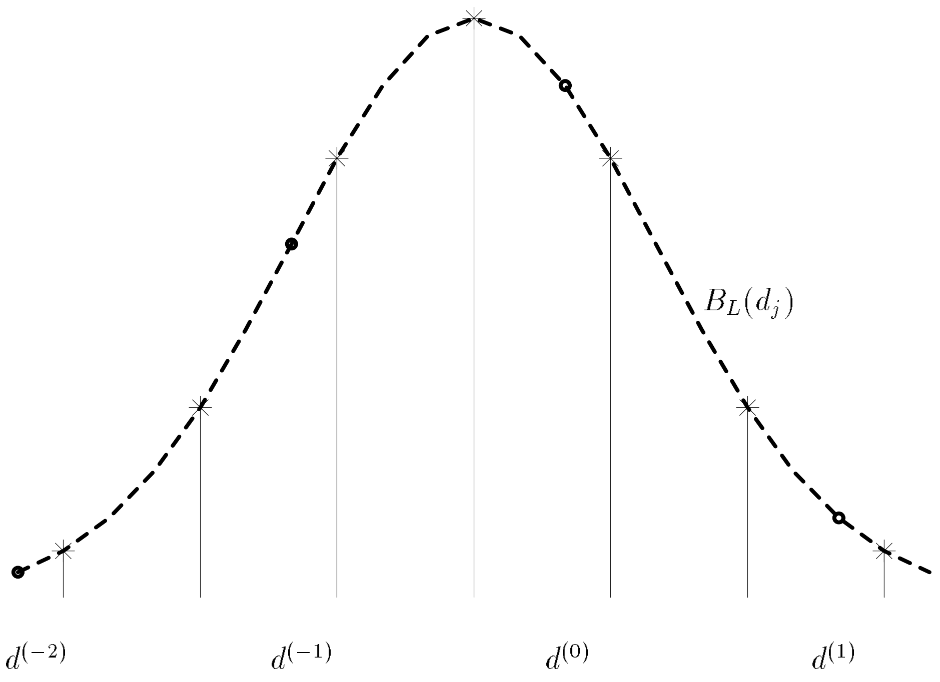

Figure 2: Disposition of the points $..., d^{(-1)}, d^{(0)}, d^{(1)}, ...$ and corresponding impulses at the binomial curve; impulses in the points $L \pm iq/2, i = 0, 1, ...$ are marked by stars.

References

[1] D.E.Knuth, *The Art of Computer Programming. Vol.3. Sorting and Searching.* NY : Adison-Wessley, 1973.

[2] H.R.Lewis and L.Danenberg, *Data Structures & Their Algorithms.* NY : HarperCollins Publishers, 1991.

[3] J.S.Vitter and W.-C.Chen, *Design and Analysis of Coalesced Hashing.* NY : Oxford University Press, 1987.

[4] G.D.Knott, "Hashing functions," *The Computer J.*, v.18, no.3, Aug.1975, pp.265-278.

[5] G.H.Gonnet and P-A.Larson, "External hashing with limited internal storage," *Journal of ACM*, v.35, no.1, Jan.1988, pp.161-184.

[6] J.P.Hayes, *Computer Architecture and Organization.* 2-nd Ed., McGraw-Hill, 1988.

[7] V.N.Koshelev, "Computational aspects of the triangle axiom," *Cybernetics and Computer Technique,* no. 4, Moscow : "Nauka", 1988, pp.204-216 (in Russian).

[8] R.G.Gallager, *Information Theory and Reliable Communication.* NY : John Wiley and Sons, 1968.

[9] T.M.Cover and J.A.Thomas, *Elements of Information Theory.* NY : John Wiley and Sons, 1991.

[10] C.C.Chang, C.Y.Chen, and J.K.Yan, "On the design of a machine-idependent perfect hashing scheme," *The Computer J.*, v.34, no.5, Oct.1991, pp.469-474.